CoROR: A COmposable Rule-entailment Owl Reasoner for Resource-Constrained Devices

Wei Tai, John Keeney, Declan O'Sullivan

Knowledge and Data Engineering Group,

School of Computer Science and Statistics, Trinity College Dublin

24/07/2011

Background and motivation

- The Semantic Sensor Network (SSN) is a recently emerged research strand using Semantic Web technologies, in particular OWL and its reasoning technologies, to solve problems encountered in traditional sensor network systems, e.g.
  - improving interoperability in heterogeneous environment,
  - enabling intelligent data processing,
  - enabling intelligent management.

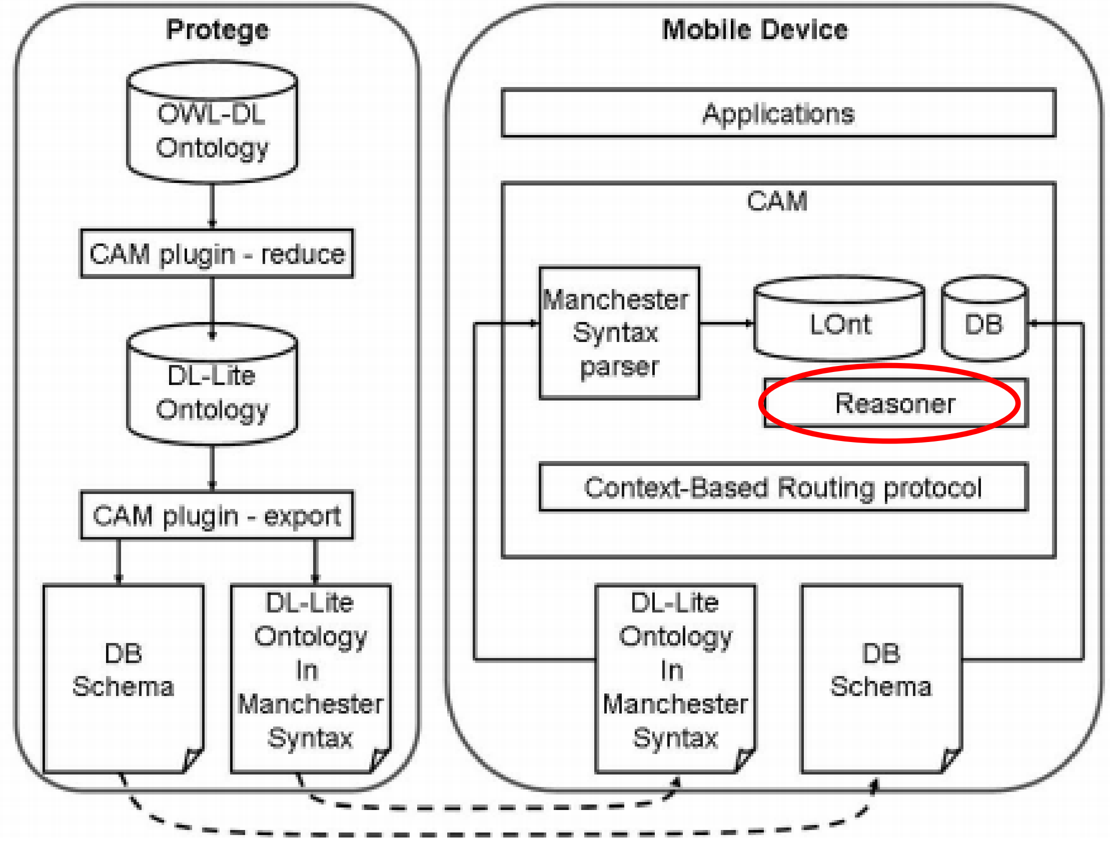
- A important research facets within SSN is the demand for “on-device” semantic reasoning, e.g.
  - Information filtering in context-aware mobile personal information system,
  - localized fault diagnosis in wireless sensor network,
  - context-addressable messaging services in mobile ad-hoc networks.

Example

- Context addressable messaging architecture
  - An OWL reasoner is used to perform address resolving.
  - Terminological data are stored in LOnt, and context data are stored in database.
  - Implemented on Nokia N800.

Further reading

M. Koziuk, J. Domaszewicz, R. Schoeneich, M. Jablonowski, and P. Boetzel, "Mobile Context-Addressable Messaging with DL-Lite Domain Model," in *Proc. European Conf. on Smart Sensing and Context* (EuroSSC'08), 2008.



Existing resource-constrained OWL reasoners

- MiRE4OWL
  - Forward-chaining RETE-based OWL reasoner,
  - Unoptimized internal reasoning algorithm,
  - C++ for PPC.
- µOR
  - Backward-chaining resolution-based OWL reasoner,
  - Scales well for large amount of instance data,
  - Suitable for small terminological box,
  - CDC compatible.
- Bossam
  - Forward-chaining RETE-based OWL reasoner,
  - Concentrated on web-friendly and distributed reasoning,
  - CDC compatible.
- Others
  - E.g. the one in MCA (J2ME but no further information on the platform), KRHyper and so on.

- They are designed for relatively “more powerful” mobile devices, e.g. mobile phone or PDA, rather than highly constrained mobile devices, e.g. sensors.

COROR: an overview

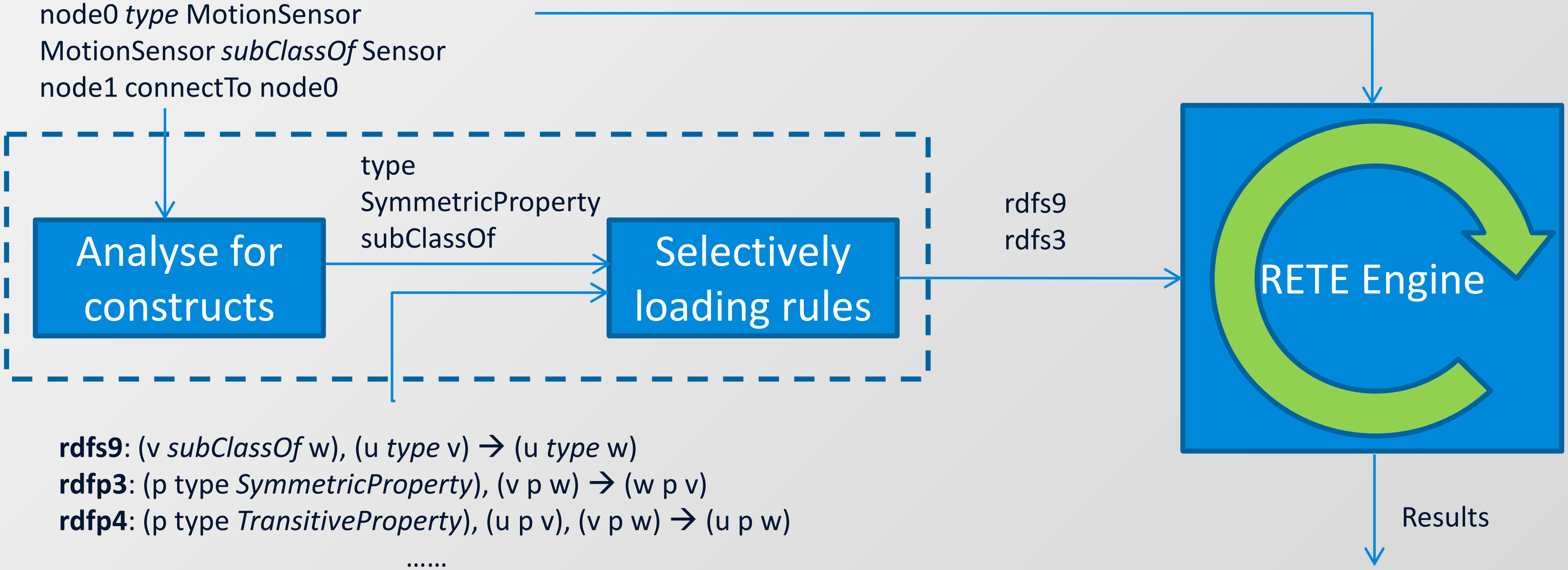
- COROR is a COmposable Rule-entailment Owl Reasoner for resource-constrained devices.
    - Forward-chaining RETE-based OWL reasoner.
    - Incorporates two novel reasoner composition algorithms to reduce the memory footprint, i.e.
        - selective rule loading algorithm, and
        - two phase RETE algorithm.

- COROR is referred to as ***composable*** as it dimensions reasoning algorithm on-the-fly according to the semantic features of the ontology to be reasoned, especially the OWL constructs.

Selective rule loading algorithm

- The selective rule loading algorithm selectively load rules into a reasoner depending on the reasoning capabilities required.
  - According to the rule-construct dependency relationship.
  - Loads only OWL inference rules required to reason the particular ontology.

connectWith *type SymmetricProperty*
node1 *type* MotionSensor
node0 *type* MotionSensor
MotionSensor *subClassOf* Sensor
node1 connectTo node0

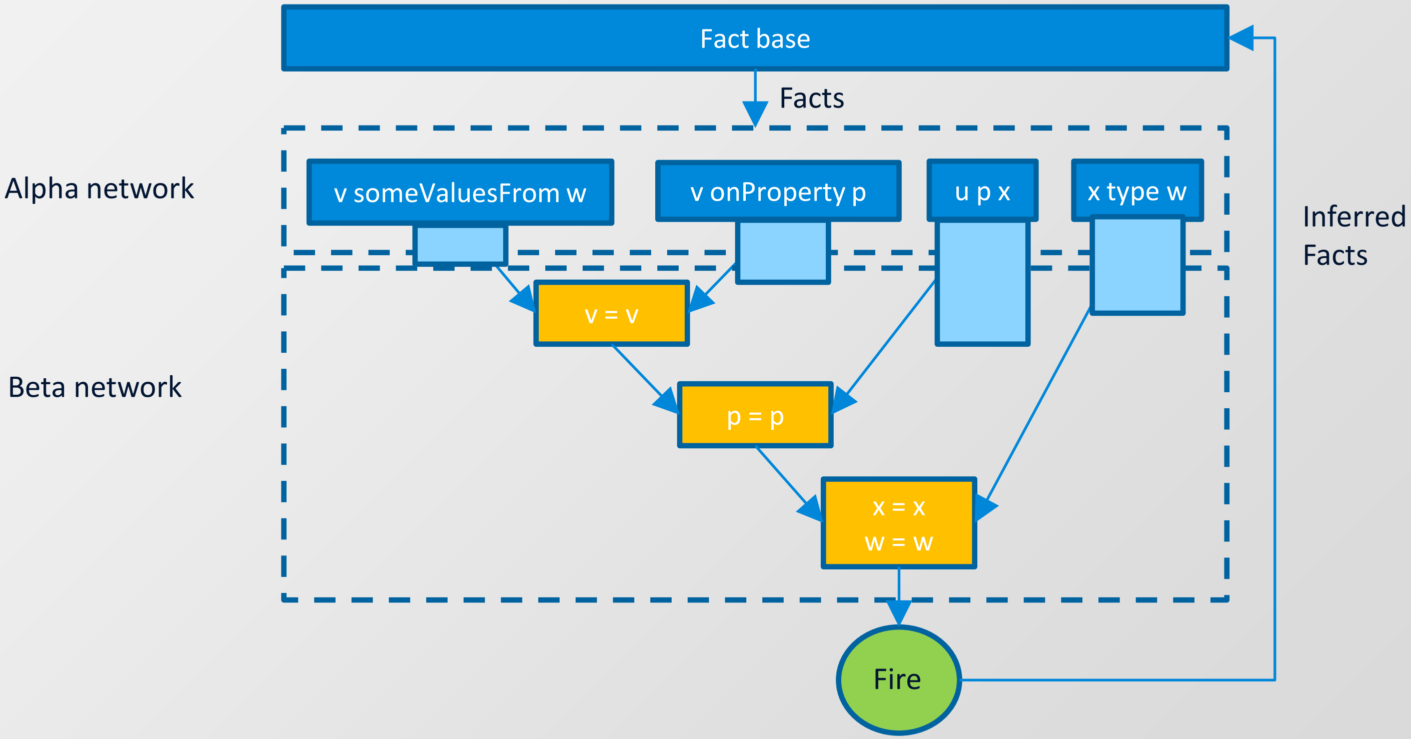


rdfs9: (v *subClassOf* w), (u *type* v) → (u *type* w)
rdfp3: (p type *SymmetricProperty*), (v p w) → (w p v)
rdfp4: (p type *TransitiveProperty*), (u p v), (v p w) → (u p w)
……

A short introduction to RETE

- RETE is a fast pattern matching algorithm for forward-chaining production systems. It performs pattern matching using a network structure termed as RETE network.

(?v owl:someValuesFrom ?w), (?v owl:onProperty ?p), (?u ?p ?x), (?x rdf:type ?w) → (?u rdf:type ?v)

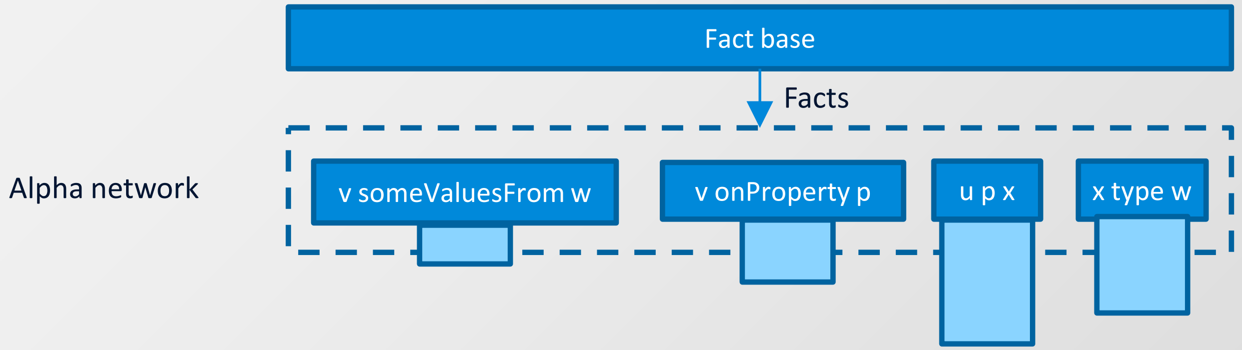


Two-phase RETE algorithm

- The two-phase RETE algorithm uses an interrupted RETE construction mechanism to build a customized RETE network.
  - The alpha network is constructed
    - Common condition node sharing
  - RETE construction is interrupted by an initial fact matching after the construction of the alpha network to collect some information about the ontology, e.g.
    - Number of partially matched facts for each condition node (in use),
    - Selectivity between two joining alpha nodes (potential),
    - and etc.
  - Rather than applied beta network optimization heuristics directly, they are applied using the collected information to construct a customized beta network, i.e.
    - Most specific condition first
      - The number of partially matched facts for each condition node is used as its specificity.
    - Connectivity

An example

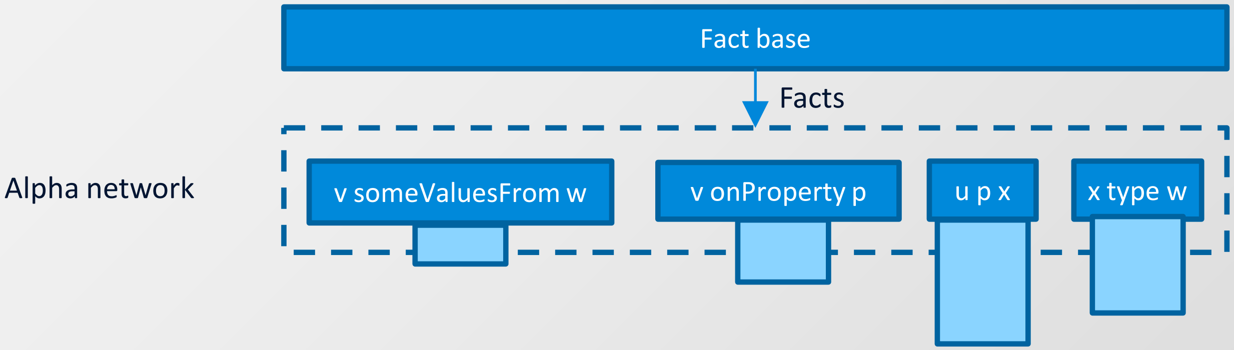
(?v owl:someValuesFrom ?w), (?v owl:onProperty ?p), (?u ?p ?x), (?x rdf:type ?w) → (?u rdf:type ?v)



Ireland's EU Structural Funds Programmes 2007 - 2013
Co-funded by the Irish Government and the European Union
EUROPEAN UNION STRUCTURAL FUNDS
HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas
AN ROINN OIDEACHAIS AGUS EOLAÍOCHTA
DEPARTMENT OF EDUCATION AND SCIENCE

An example

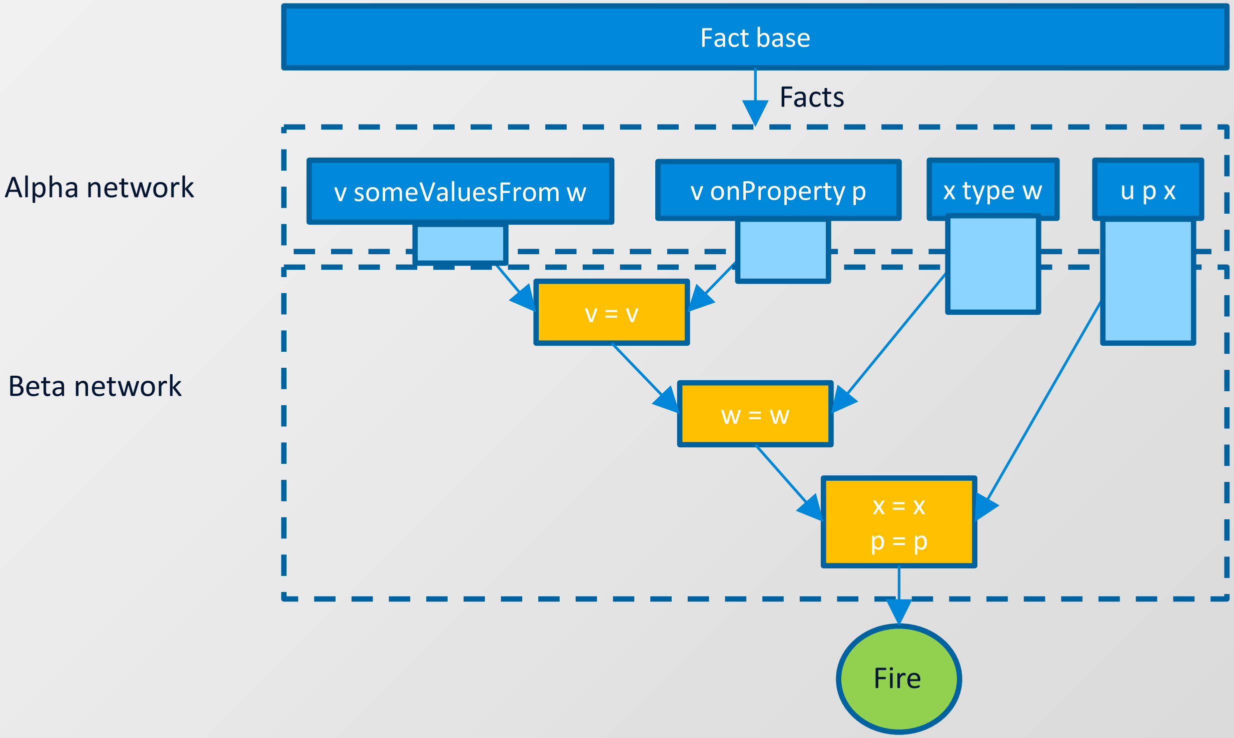
(?v owl:someValuesFrom ?w), (?v owl:onProperty ?p), (?u ?p ?x), (?x rdf:type ?w) → (?u rdf:type ?v)



Ireland's EU Structural Funds Programmes 2007 - 2013
Co-funded by the Irish Government and the European Union
EUROPEAN UNION STRUCTURAL FUNDS
HEA
Higher Education Authority
An tÚdarás um Ard-Oideachas
AN ROINN OIDEACHAIS AGUS EOLAÍOCHTA
DEPARTMENT OF EDUCATION AND SCIENCE

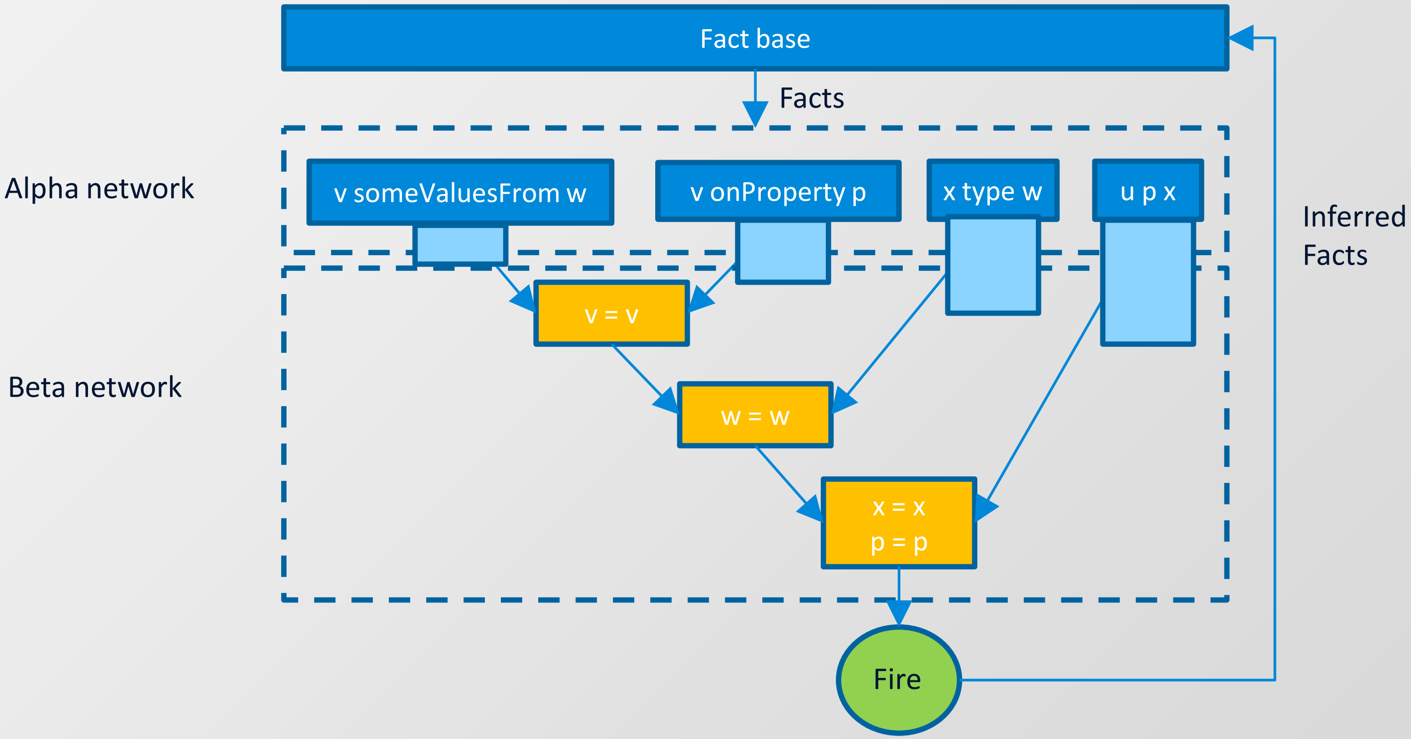
An example

(?v owl:someValuesFrom ?w), (?v owl:onProperty ?p), (?u ?p ?x), (?x rdf:type ?w) → (?u rdf:type ?v)



An example

(?v owl:someValuesFrom ?w), (?v owl:onProperty ?p), (?u ?p ?x), (?x rdf:type ?w) → (?u rdf:type ?v)



Implementation

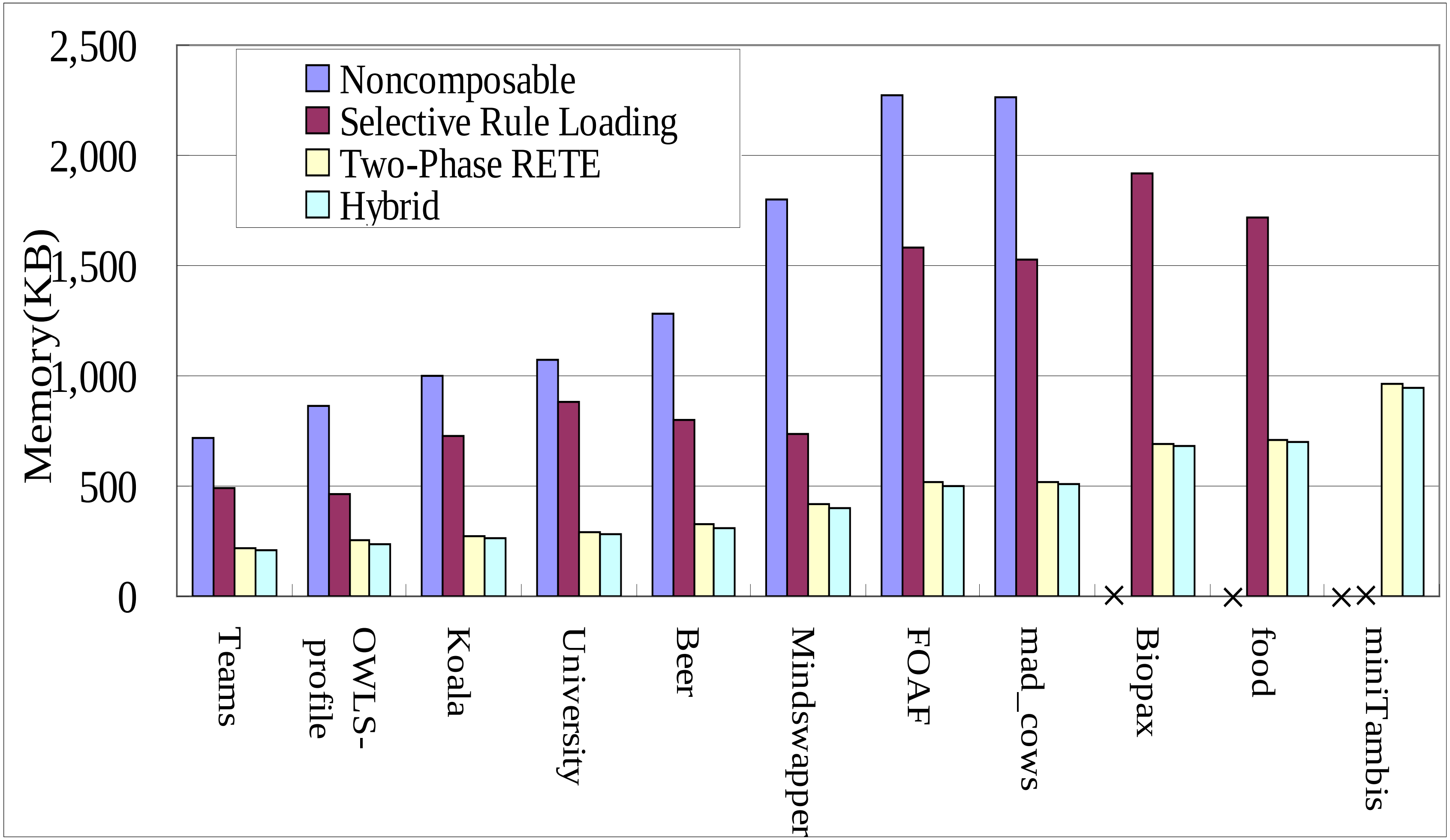
- COROR is implemented
  - In J2ME
    - CLDC 1.1 conformant.
  - Based on μJena,
    - Enabling owl ontology reading, parsing and manipulation,
    - Requiring a volume of code refactoring as μJena does not originally support for owl reasoning.
  - On Sun SPOT emulator.
  - Using entailments and atomic query as key reasoning tasks, combining which are enough to simulate common reasoning tasks.

- Four composition modes
  - Non-composable mode (Original RETE)
  - Selective rule loading mode
  - Two-phase RETE mode
  - Hybrid mode

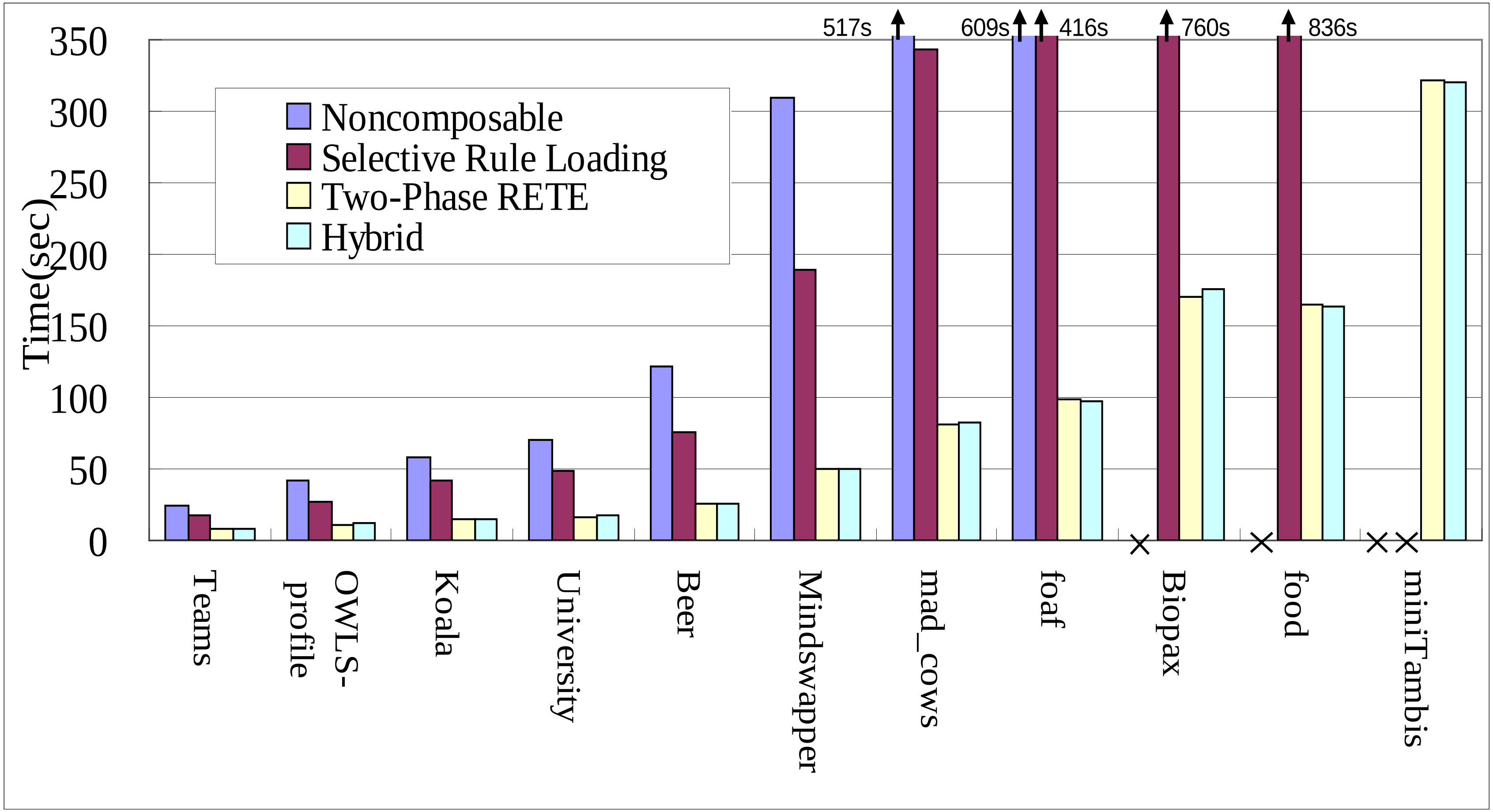
Experiment design

- Metrics
  - Memory
  - Reasoning time
- Experiments include
  - Evaluation and comparison between different composition modes of COROR.
    - Performed on Sun SPOT emulator.
  - Evaluation and comparison between COROR and other reasoners, i.e.
    - Jena, Bossam (a mobile reasoner, CDC only), BaseVISor (time only), OWLIM, and Pellet.
    - MiRE4OWL and µOR are not accessible.
    - Performed on desktop due to the other reasoners cannot run on Sun SPOT.
- Selected ontology include 17 well-known ontologies from different domains
  - Relatively free of error
  - Small to medium sized
  - Avoid over-/under- use of some owl constructs

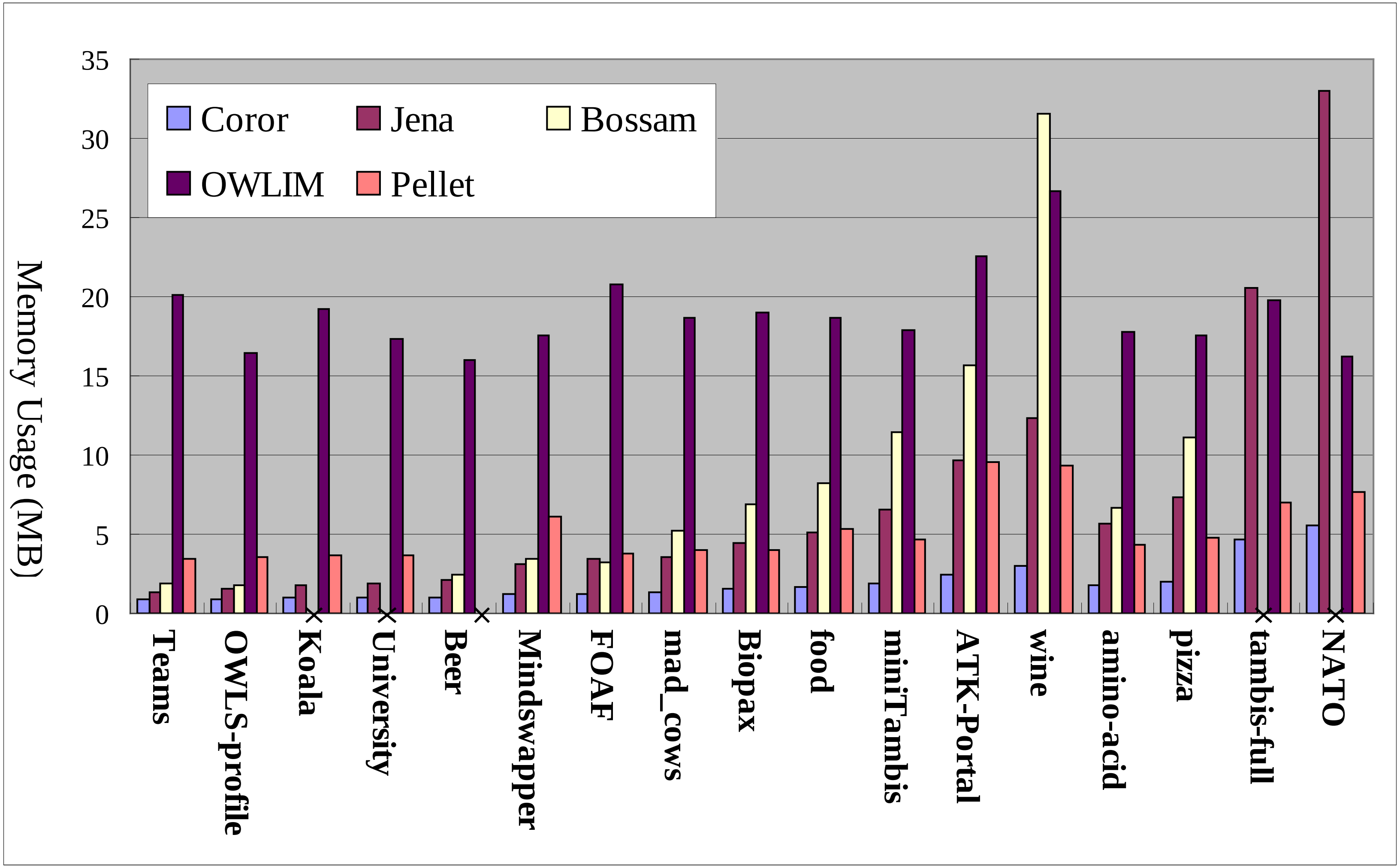
Experiment results (memory-1)



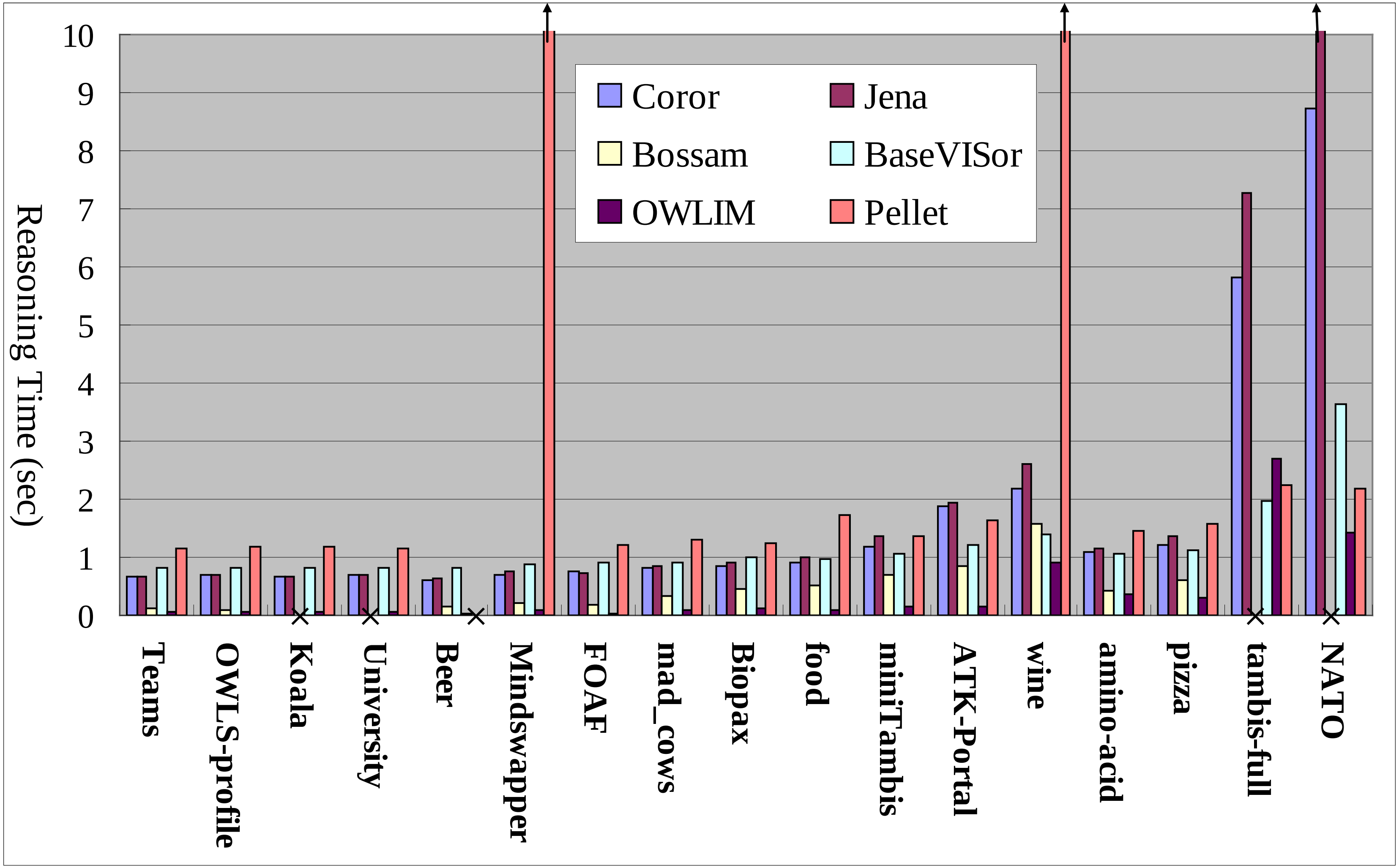
Experiment results (time-1)



Experiment results (memory-2, Coror-hybrid)



Experiment results (time-2, Coror-hybrid)



Discussion based on the empirical results

- All composable modes require less memory and reasoning time than the noncomposable mode
  - The two-phase RETE and Hybrid uses a lot less memory than the other modes as for this rule set the two-phase RETE algorithm can optimize the RETE structure as if unused rules are “removed” so the hybrid mode (combining the two-phase RETE algorithm and the selective rule loading algorithm) does not gain much more memory/time reduction comparing to the two-phase RETE algorithm.

- Use the least memory among all evaluated reasoners while have reasoning time comparable to Jena forward chaining reasoner.
  - For small ontology (which are expected to be applied on sensors) COROR uses much less memory than the others.

Conclusion and future work

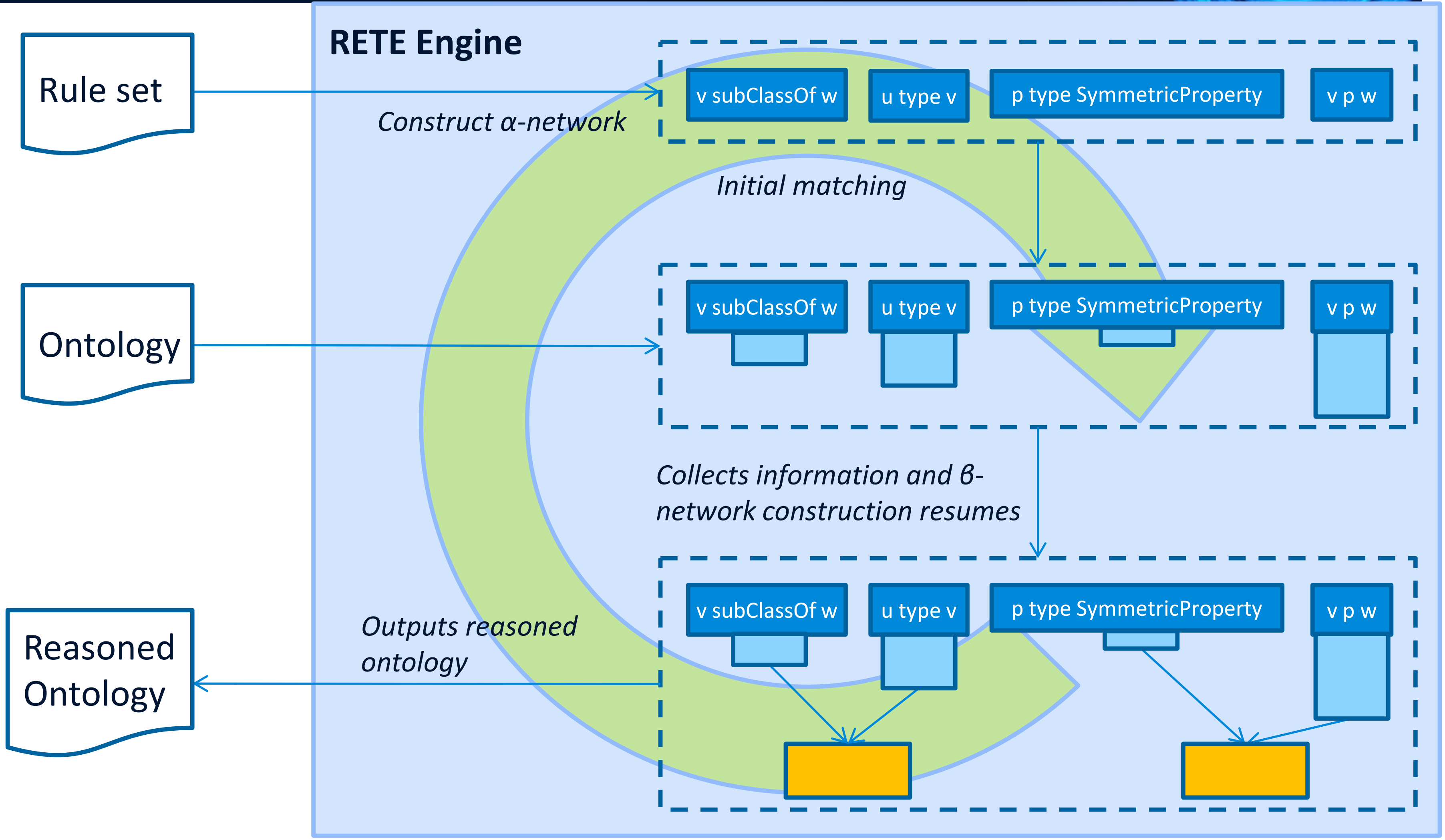
- Conclusion
  - Motivation
    - To enable intelligence on the edge of sensor network.
  - State of the art mobile OWL reasoners
  - COROR the composable reaoner
    - Two composition algorithms: the selective rule loading algorithm and the two-phase RETE algorithm.
  - Experiments design and results
    - Two experiments
    - Results
      - COROR uses much less memory without sacrificing time performance.

- Future work includes
  - More heuristics can be applied during the RETE network construction phase, and more information can be collected.
  - Support conjunctive query languages e.g. SPARQL.
  - OWL 2 support.

Thank you

Questions?

Two-phase RETE algorithm (cont'd)



An analytical comparison between composition algorithms

- Reasoning algorithm independence:
  - Selective rule loading algorithm: applicable on all rule-based reasoning algorithm, does not require changes in the reasoning algorithm itself, and relatively easy to implement.
  - Two-phase rete algorithm: applicable only on RETE algorithm, require change in RETE, and relatively hard to implement.
- Semantic independence:
  - They both are semantic independent.
- Flexibility in handling changes:
  - Addition can be handled incrementally by the two-phase RETE algorithm however may introduce unseen owl constructs requiring the re-execution of the selective rule loading algorithm.
  - Simple deletion may cause logical inconsistency so re-execution is required for both algorithms.